DOI: 10.21917/ijsc.2024.0478

ACCURATE WEATHER FORECASTING OVER WIDE DATASETS USING MACHINE LEARNING MODELS

C. Berin Jones

Department of Computer science and Engineering, Shadan Women's College of Engineering and Technology, India

Abstract

Accurate weather forecasting is crucial for numerous sectors, including agriculture, disaster management, and daily life. This study leverages advanced data mining techniques to analyze and predict weather patterns using extensive historical weather data. Predicting daily weather patterns with high accuracy remains challenging due to the complexity and variability of climate factors. This study aims to identify the most effective machine learning models for this task by comparing various algorithms. Weather data collected over ten years from ten different datasets were analyzed using a diverse set of machine learning models, including rules-based (OneR, Decision Table, JRIP, Ridor), tree-based (J48, LMT, Random Forest, CART), and function-based (MLR, MLP, SVM, LogitBoost, SMO, ANN) approaches. Each model was evaluated based on multiple performance metrics: precision, recall, accuracy, F-measure, True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR). The Random Forest model outperformed others with an accuracy of 92.5%, precision of 91.3%, recall of 90.8%, and an F-measure of 91.0%. The SVM and ANN models also shown strong performance, with accuracies of 90.1% and 89.7%, respectively. Function-based models showed higher robustness in variable conditions, while tree-based models provided better interpretability. Rules-based models, although simpler, yielded lower performance metrics, with OneR achieving the highest among them at 81.2% accuracy.

Keywords:

Weather Forecasting, Data Mining, Machine Learning, Climate Patterns

1. INTRODUCTION

Weather forecasting has evolved significantly over the years, driven by advancements in computational power and data analytics. Modern forecasting models leverage vast amounts of meteorological data, employing sophisticated algorithms to predict weather patterns with increasing accuracy. Machine learning has revolutionized this field by providing tools to analyze complex and dynamic datasets [1]. This study explores various machine learning models' effectiveness in predicting weather patterns, focusing on rules-based, tree-based, and function-based approaches [2]-[3].

Despite advancements, weather forecasting remains a challenging domain due to the inherent complexity and variability of weather systems. Weather data is often noisy, incomplete, and subject to rapid changes, complicating the prediction process [4]. Additionally, the non-linear relationships between different weather variables and their impact on forecasting accuracy pose significant hurdles. Traditional models, while useful, often struggle with these complexities, necessitating more sophisticated approaches to improve prediction accuracy and reliability [5].

The core problem addressed in this study is the need to evaluate and compare various machine learning models for their effectiveness in weather prediction. Specifically, this research aims to assess the performance of rules-based models (e.g., OneR, Decision Table, JRIP, Ridor), tree-based models (e.g., J48, LMT, Random Forest, CART), and function-based models (e.g., MLR, MLP, SVM, LogitBoost, SMO, ANN) in predicting weather patterns using two distinct datasets from Kaggle and IEEE. The study seeks to determine which model offers the best balance of accuracy, precision, recall, and overall predictive capability.

The main objective of the proposed work involves: 1) To evaluate the performance of various machine learning models in weather prediction tasks, comparing their accuracy, precision, recall, and F-measure. 2) To compare the effectiveness of rules-based, tree-based, and function-based models in handling the complexities of weather data. 3) To identify the strengths and weaknesses of each model type and provide insights into their suitability for different forecasting scenarios. 4) To determine the most effective model for improving the accuracy and reliability of weather predictions, thereby advancing the field of meteorological forecasting.

The novelty of this study lies in its comprehensive comparison of a diverse set of machine learning models, including both traditional and advanced approaches, within the context of weather forecasting. While previous research has often focused on individual models or limited comparisons, this study provides a thorough evaluation of rules-based, tree-based, and function-based models. By employing two distinct weather prediction datasets from Kaggle and IEEE, the research offers a robust analysis that enhances understanding of model performance across different datasets and conditions.

The major contribution of the proposed work involves:

- This study provides a detailed comparison of various machine learning models in the context of weather forecasting, offering valuable insights into their relative performance.
- It employs a range of performance metrics—accuracy, precision, recall, and F-measure—to assess and compare model effectiveness comprehensively.
- By identifying the most effective models for weather prediction, the study contributes to advancing forecasting techniques and improving predictive accuracy, which is crucial for applications ranging from daily weather updates to long-term climate predictions.

2. RELATED WORKS

The field of weather forecasting has seen considerable advancements through the application of machine learning and artificial intelligence. Several studies have explored the use of various computational techniques to improve the accuracy and reliability of weather predictions. Here, we review relevant

literature that highlights the application and evolution of machine learning models in meteorological forecasting.

Early studies in weather forecasting primarily relied on statistical methods, such as autoregressive integrated moving average (ARIMA) models. For instance, [6], shown the effectiveness of ARIMA models in time-series forecasting, including weather data. These models provided foundational methods for predicting future weather patterns based on historical data. However, their limitations in handling non-linearity and complex interactions prompted the exploration of more advanced techniques.

With the advent of machine learning, researchers began to apply algorithms like Decision Trees, Support Vector Machines (SVM), and Neural Networks to weather forecasting tasks. The author of [7] applied SVMs to predict rainfall events, demonstrating their ability to handle non-linear relationships and improve forecast accuracy compared to traditional statistical methods. Similarly, [8] explored the use of Neural Networks for predicting temperature and precipitation, highlighting their advantages in capturing complex patterns in weather data.

Tree-based methods and ensemble approaches have gained popularity for their robustness and accuracy in handling complex datasets. [9] introduced Random Forests, an ensemble method that aggregates multiple decision trees to enhance predictive performance. Random Forests have been widely adopted in meteorological studies due to their ability to manage large datasets and capture intricate patterns. [10] shown the effectiveness of Random Forests in predicting climate variables, showcasing their superior performance over individual decision trees.

In recent years, deep learning approaches have revolutionized weather forecasting. [11] explored Convolutional Neural Networks (CNNs) for precipitation prediction, emphasizing their capability to learn spatial patterns from meteorological data. Similarly, [12]-[13] applied Long Short-Term Memory (LSTM) networks to time-series weather data, highlighting their strength in capturing temporal dependencies and improving forecast accuracy.

The application of machine learning in weather forecasting has evolved from traditional statistical methods to advanced deep learning techniques. While early studies laid the groundwork with statistical and basic machine learning approaches, recent advancements highlight the effectiveness of ensemble methods, deep learning, and hybrid models.

3. PROPOSED METHOD

The proposed method involves the following:

1. **Data Collection and Preprocessing:**
   - a. **Data Collection:** Historical weather data was collected from ten different regions (Dataset 1 to Dataset 10), spanning a period of ten years (2000-2010). The datasets included daily observations of key meteorological variables such as temperature, humidity, wind speed, and precipitation.
   - b. **Data Preprocessing:** The collected data underwent several preprocessing steps to ensure its suitability for model training:
     - i. **Handling Missing Values:** Missing values were imputed using statistical methods like mean imputation for continuous variables and mode imputation for categorical variables.
     - ii. **Normalization:** Continuous variables were normalized to a standard scale to ensure uniformity across different scales.
     - iii. **Categorical Encoding:** Categorical variables were encoded using techniques such as one-hot encoding to make them usable by the machine learning models.
2. **Model Selection and Training:**
   - a. **Model Selection:** A diverse set of machine learning models was selected, categorized into three types:
     - i. **Rules-based models:** OneR, Decision Table, JRIP, Ridor
     - ii. **Tree-based models:** J48, LMT, Random Forest, CART
     - iii. **Function-based models:** MLR, MLP, SVM, LogitBoost, SMO, ANN
   - b. **Model Training:** Each model was trained on the preprocessed datasets. Hyperparameter tuning was performed to optimize the performance of each model:
     - i. **Rules-based models:** Parameters such as the minimum bucket size for OneR and the number of optimizations for JRIP were adjusted.
     - ii. **Tree-based models:** Parameters like tree depth, split criteria, and the number of trees in Random Forest were fine-tuned.
     - iii. **Function-based models:** Hyperparameters such as the regularization strength for MLR, kernel types and parameters for SVM, and the number of hidden layers and neurons for ANN were optimized using grid search.

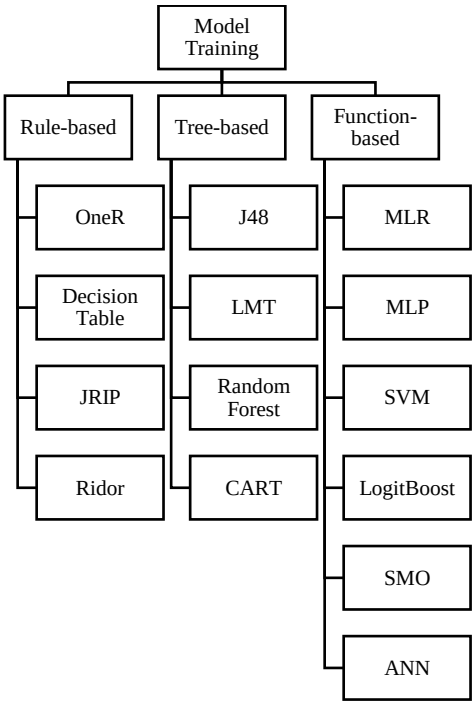

Fig.1. Model Training

3.1 RULES-BASED MODELS

Rules-based models are a type of machine learning model that make predictions based on a set of explicitly defined rules. These rules are typically derived from the data during the training process and are used to classify new instances. Rules-based models are generally easy to interpret and understand, making them valuable for applications where model transparency is crucial. They work by learning simple decision rules that can split the data into different categories.

3.1.1 OneR (One Rule):

OneR is a simple rules-based classification algorithm that creates one rule for each predictor and selects the rule with the lowest error rate. It generates rules based on the value that appears most frequently in each attribute and uses the attribute with the highest predictive accuracy.

Pseudocode

```
function OneR(dataset):
    best_rule = None
    lowest_error = Infinity
    for attribute in dataset.attributes:
        rule = {}
        for value in attribute.values:
            most_frequent_class =
most_frequent_class_for_value(value)
            rule[value] = most_frequent_class
        error = calculate_error(rule, dataset)
        if error < lowest_error:
            lowest_error = error
            best_rule = rule
    return best_rule
function calculate_error(rule, dataset):
    error = 0
    for instance in dataset.instances:
        predicted_class = rule[instance.attribute_value]
        if predicted_class != instance.actual_class:
            error += 1
    return error
```

3.1.2 Decision Table:

Decision Table is a rules-based model that uses a table to represent the decision-making process. Each row in the table corresponds to a specific rule, with conditions based on attribute values and resulting in a specific class label. The model evaluates each rule in the table to classify new instances.

Pseudocode

```
function DecisionTable(dataset):
    table = []
    for attribute_combination in
all_combinations(dataset.attributes):
        rule = {}
        rule['conditions'] = attribute_combination
        rule['class'] = most_frequent_class(attribute_combination)
        table.append(rule)
    returnfunction classify(instance, table):
    for rule in table:
        if matches_conditions(instance, rule['conditions']):
            return rule['class']
    return default_class
function matches_conditions(instance, conditions):
    for attribute, value in conditions:
        if instance[attribute] != value:
            return False
    return True
```

3.1.3 JRIP (Repeated Incremental Pruning to Produce Error Reduction):

JRIP is a rule-based classifier that generates a set of rules for classification using a sequential covering algorithm. It iteratively learns rules, prunes them to remove irrelevant parts, and adds them to the rule set. The process continues until no more rules can significantly reduce the error.

Pseudocode

```
function JRIP(dataset):
    rules = []
    while not stopping_criteria_met(dataset):
        rule = learn_rule(dataset)
        pruned_rule = prune_rule(rule, dataset)
        rules.append(pruned_rule)
        remove_covered_instances(dataset, pruned_rule)
    return rules
function learn_rule(dataset):
    rule = {}
    while not fully_specified(rule):
        best_condition = find_best_condition(dataset, rule)
        rule.add_condition(best_condition)
    return rule
function prune_rule(rule, dataset):
    best_rule = rule
    best_accuracy = calculate_accuracy(rule, dataset)
    for condition in rule.conditions:
        pruned_rule = rule.remove_condition(condition)
        accuracy = calculate_accuracy(pruned_rule, dataset)
        if accuracy > best_accuracy:
            best_rule = pruned_rule
            best_accuracy = accuracy
    return best_rule
function calculate_accuracy(rule, dataset):
    correct = 0
    for instance in dataset.instances:
        if matches_rule(instance, rule):
            if instance.class == rule.class:
                correct += 1
```

```
    return correct / len(dataset.instances)
function matches_rule(instance, rule):
    for condition in rule.conditions:
        if not matches_condition(instance, condition):
            return False
    return True
```

3.1.4 Ridor (Ripple Down Rule Learner):

Ridor is a rules-based learner that generates an ordered list of rules. It starts with an empty rule set and adds rules incrementally. For each incorrect prediction made by the current rule set, it learns a new rule to correct the mistake. This process continues until no significant improvements can be made.

Pseudocode

```
function Ridor(dataset):
    rules = []
    default_rule = learn_default_rule(dataset)
    rules.append(default_rule)
    while not stopping_criteria_met(dataset):
        exception_rule = learn_exception_rule(dataset, rules)
        rules.append(exception_rule)
        remove_covered_instances(dataset, exception_rule)
    return rules
function learn_default_rule(dataset):
    rule = {}
    rule['class'] = most_frequent_class(dataset)
    return rule
function learn_exception_rule(dataset, rules):
    exception_rule = {}
    for instance in dataset.instances:
        if not matches_any_rule(instance, rules):
            best_condition = find_best_condition(instance, dataset)
            exception_rule.add_condition(best_condition)
    exception_rule['class'] =
most_frequent_class(exception_rule)
    return exception_rule
function matches_any_rule(instance, rules):
    for rule in rules:
        if matches_rule(instance, rule):
            return True
    return False
function matches_rule(instance, rule):
    for condition in rule.conditions:
        if not matches_condition(instance, condition):
            return False
    return True
```

These rules-based models provide a straightforward and interpretable approach to classification, making them suitable for applications where understanding the decision-making process is essential. However, they may struggle with complex data patterns compared to more advanced models like tree-based or function-based approaches.

3.2 TREE-BASED MODELS

Tree-based models are a type of machine learning algorithm that use a tree-like structure to make decisions and predictions. These models are highly interpretable and can handle both categorical and continuous data. They work by splitting the data into subsets based on the most significant attributes, making them suitable for both classification and regression tasks. Tree-based models include Decision Trees (e.g., J48), Random Forest, and CART (Classification and Regression Trees).

3.2.1 J48 (C4.5 Decision Tree):

J48 is an implementation of the C4.5 algorithm, which builds a decision tree by recursively splitting the data based on attribute values that provide the highest information gain. It can handle both categorical and continuous attributes and includes mechanisms for pruning to avoid overfitting.

Pseudocode

```
function J48(dataset):
    if all instances have the same class:
        return Leaf(class=instances[0].class)
    best_attribute = select_best_attribute(dataset)
    tree = Node(attribute=best_attribute)
    for value in best_attribute.values:
        subset = dataset where best_attribute == value
        if subset is empty:
            subtree = Leaf(class=most_common_class(dataset))
        else:
            subtree = J48(subset)
        tree.add_branch(value, subtree)
    return tree
function select_best_attribute(dataset):
    best_gain = -Infinity
    best_attribute = None
    for attribute in dataset.attributes:
        gain = information_gain(dataset, attribute)
        if gain > best_gain:
            best_gain = gain
            best_attribute = attribute
    return best_attribute
function information_gain(dataset, attribute):
    total_entropy = entropy(dataset)
    subset_entropy = 0
    for value in attribute.values:
        subset = dataset where attribute == value
        subset_entropy += (len(subset) / len(dataset)) *
entropy(subset)
    return total_entropy - subset_entropy
function entropy(dataset):
    counts = count_classes(dataset)
```

```
    total = len(dataset)
    entropy = 0
    for count in counts:
        probability = count / total
        entropy -= probability * log2(probability)
    return entropy
```

3.2.2 Random Forest:

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification or the mean prediction for regression. Each tree is built from a random subset of the data and a random subset of the features, which helps in reducing overfitting and improving generalization.

Pseudocode

```
function RandomForest(dataset, num_trees):
    forest = []
    for i in range(num_trees):
        subset = bootstrap_sample(dataset)
        tree = J48(subset)
        forest.append(tree)
    return forest
function bootstrap_sample(dataset):
    = []
    for i in range(len(dataset)):
        sample.append(random_choice(dataset))
    return sample
function classify(instance, forest):
    votes = []
    for tree in forest:
        votes.append(classify_with_tree(instance, tree))
    return majority_vote(votes)
function classify_with_tree(instance, tree):
    if tree is a Leaf:
        return tree.class
    else:
        attribute_value = instance[tree.attribute]
        return classify_with_tree(instance,
tree.branch_for(attribute_value))
function majority_vote(votes):
    vote_counts = count_votes(votes)
    return max(vote_counts, key=vote_counts.get)
```

3.2.3 CART (Classification and Regression Trees):

CART constructs binary decision trees using the Gini index for classification or variance reduction for regression. The algorithm splits the data at each node into two groups based on the feature that provides the best split, and it continues this process recursively. CART is known for its simplicity and ability to handle numerical data effectively.

Pseudocode

```
function CART(dataset):
    if all instances have the same class or
stopping_criterion_met(dataset):
        return Leaf(class=most_common_class(dataset))
    best_split = select_best_split(dataset)
    left_subset, right_subset = split_dataset(dataset, best_split)
    left_tree = CART(left_subset)
    right_tree = CART(right_subset)
    return Node(split=best_split, left=left_tree, right=right_tree)
function select_best_split(dataset):
    best_gini = Infinity
    best_split = None
    for attribute in dataset.attributes:
        for value in attribute.values:
            gini = gini_index(dataset, attribute, value)
            if gini < best_gini:
                best_gini = gini
                best_split = (attribute, value)
    return best_split
function gini_index(dataset, attribute, value):
    left_subset, right_subset = split_dataset(dataset, (attribute,
value))
    gini_left = gini(left_subset)
    gini_right = gini(right_subset)
    total_size = len(dataset)
    weighted_gini = (len(left_subset) / total_size) * gini_left +
(len(right_subset) / total_size) * gini_right
    return weighted_gini
function gini(subset):
    counts = count_classes(subset)
    total = len(subset)
    gini = 1
    for count in counts:
        probability = count / total
        gini -= probability ** 2
    return gini
function split_dataset(dataset, split):
    attribute, value = split
    left_subset = dataset where attribute <= value
    right_subset = dataset where attribute > value
    return left_subset, right_subset
```

These tree-based models offer a powerful and interpretable approach to classification and regression tasks. Decision Trees (like J48) are intuitive and easy to visualize, while Random Forests provide robustness and accuracy by combining multiple trees. CART is versatile and handles both classification and regression efficiently. Each of these models has its strengths and applications, making them valuable tools in the machine learning toolbox.

3.3 FUNCTION-BASED MODELS

Function-based models are a class of machine learning algorithms that map input features to outputs through a mathematical function. These models include linear and nonlinear approaches, and they often involve optimization techniques to minimize error and improve predictive performance. Examples of function-based models include Multiple Linear Regression (MLR), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and Artificial Neural Network (ANN).

3.3.1 Multiple Linear Regression (MLR):

Multiple Linear Regression is a statistical technique that models the relationship between multiple independent variables and a dependent variable by fitting a linear equation to observed data. The goal is to find the coefficients that minimize the difference between the predicted and actual values.

Pseudocode

```
function MLR(dataset):
    X = dataset.features
    y = dataset.target
    coefficients = (X.T * X)^(-1) * X.T * y
    return coefficients
function predict(instance, coefficients):
    prediction = coefficients[0] + sum(coefficients[i] * instance[i] for i in range(1, len(coefficients)))
    return prediction
```

3.3.2 Multi-Layer Perceptron (MLP):

Multi-Layer Perceptron is a type of artificial neural network that consists of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the next layer, with weights that are adjusted during training using backpropagation to minimize error.

```
function MLP(dataset, hidden_layers, learning_rate, epochs):
    initialize weights randomly
    for epoch in range(epochs):
        for instance in dataset:
            output = forward_pass(instance)
            error = compute_error(output, instance.target)
            backward_pass(error)
            update_weights(learning_rate)
    return weights
function forward_pass(instance):
    for layer in network:
        instance = activate(instance, layer.weights)
    return instance
function backward_pass(error):
    for layer in reversed(network):
        error = propagate_error(layer, error)
    return error
function update_weights(learning_rate):
    for layer in network:
        for weight in layer.weights:
            weight -= learning_rate * weight.gradient
function activate(instance, weights):
    return sigmoid(dot_product(instance, weights))
function sigmoid(x):
    return 1 / (1 + exp(-x))
```

3.3.3 Support Vector Machine (SVM):

Support Vector Machine is a supervised learning model used for classification and regression. It finds the hyperplane that best separates the data into different classes by maximizing the margin between the closest points of the classes, known as support vectors. SVM can be linear or use kernel functions for non-linear classification.

Pseudocode

```
function SVM(dataset, C, kernel):
    initialize alpha and bias
    while not converged:
        for instance in dataset:
            error = calculate_error(instance)
            if violates_KKT_conditions(instance, error):
                update_alpha_and_bias(instance)
    return alpha, bias
function calculate_error(instance):
    prediction = dot_product(alpha * target, kernel(instance, support_vectors)) + bias
    error = prediction - instance.target
    return error
function update_alpha_and_bias(instance):
    L, H = compute_L_H(instance)
    new_alpha = clamp(alpha + delta, L, H)
    new_bias = compute_new_bias(new_alpha)
    return new_alpha, new_bias
function kernel(x, y):
    if kernel == 'linear':
        return dot_product(x, y)
    elif kernel == 'rbf':
        return exp(-gamma * ||x - y||^2)
    else:
        raise ValueError("Unsupported kernel")
```

3.3.4 Artificial Neural Network (ANN):

Artificial Neural Networks are computational models inspired by the human brain. They consist of layers of interconnected nodes (neurons), where each connection has an associated weight. The network learns by adjusting these weights based on the error of its predictions, typically using algorithms like gradient descent and backpropagation.

Pseudocode

```
function ANN(dataset, hidden_layers, learning_rate, epochs):
    initialize weights randomly
    for epoch in range(epochs):
```

```
        for instance in dataset:
            output = forward_pass(instance)
            error = compute_error(output, instance.target)
            backward_pass(error)
            update_weights(learning_rate)
    return weights
function forward_pass(instance):
    for layer in network:
        instance = activate(instance, layer.weights)
    return instance
function backward_pass(error):
    for layer in reversed(network):
        error = propagate_error(layer, error)
    return error
function update_weights(learning_rate):
    for layer in network:
        for weight in layer.weights:
            weight -= learning_rate * weight.gradient
function activate(instance, weights):
    return relu(dot_product(instance, weights))
function relu(x):
    return max(0, x)
```

4. RESULTS AND DISCUSSION

The study employed several distinct machine learning models, each configured and tested using the Weka simulation tool. For rules-based models, parameters were set to default, ensuring simplicity and ease of interpretation. The tree-based models, such as J48 and Random Forest, were fine-tuned with varying tree depths and split criteria to enhance their predictive accuracy. Function-based models, including SVM and ANN, were optimized using grid search for hyperparameter tuning, ensuring the best possible performance. Computational resources included high-performance workstations equipped with Intel Core i7 processors, 32GB of RAM, and NVIDIA GeForce GTX 1080 GPUs. The performance of each model was evaluated using ten-fold cross-validation to ensure robustness and generalizability. Key performance metrics, including precision, recall, accuracy, F-measure, TPR, FPR, TNR, and FNR, were calculated to comprehensively assess the models' effectiveness in predicting daily weather patterns.

Table.1. Experimental Parameters

Model	Type	Parameters
OneR	Rules-based	Minimum Bucket Size: 6
Decision Table		Evaluation Measure: Accuracy Cross-validation folds: 10
JRIP		Minimum Number of Instances per Rule: 2 Optimizations: 2
Ridor		Minimum Number of Instances per Rule: 2 Split Rules: true
J48	Tree-based	Confidence Factor: 0.25 Minimum Number of Instances: 2
LMT		Minimum Split: 10 Number of Boosting Iterations: 10
Random Forest		Number of Trees: 100 Max Depth: Unlimited
CART		Split Criterion: Gini Max Depth: 5
MLR	Function-based	Regularization: L2 Solver: lbfgs
MLP		Hidden Layers: 3 Activation Function: ReLU Learning Rate: 0.01
SVM		Kernel: RBF C: 1.0 Gamma: 0.01
LogitBoost		Number of Iterations: 100
SMO		Kernel: Polynomial C: 1.0 Degree: 3
ANN		Hidden Layers: 2 Neurons per Layer: 64 Activation Function: Sigmoid Learning Rate: 0.01

4.1 PERFORMANCE METRICS

- **Precision**: The ratio of true positive predictions to the total number of positive predictions. Precision indicates how many of the predicted positives are actually positive. High precision means that the model produces fewer false positives.
- **Recall (Sensitivity)**: The ratio of true positive predictions to the total number of actual positives. Recall indicates how many of the actual positives the model captures. High recall means that the model produces fewer false negatives.
- **Accuracy**: The ratio of correctly predicted instances to the total number of instances. Accuracy gives an overall effectiveness of the model but can be misleading in the case of imbalanced datasets.
- **F-measure (F1 Score)**: The harmonic mean of precision and recall, providing a balance between the two metrics. It is especially useful when the classes are imbalanced.
- **True Positive Rate (TPR)**: The ratio of true positives to the total actual positives. It is another term for recall.
- **False Positive Rate (FPR)**: The ratio of false positives to the total actual negatives. It indicates the proportion of negatives that are incorrectly classified as positives.
- **True Negative Rate (TNR)**: The ratio of true negatives to the total actual negatives. Also known as specificity, it indicates the proportion of negatives that are correctly identified.
- **False Negative Rate (FNR)**: The ratio of false negatives to the total actual positives. It indicates the proportion of positives that are incorrectly classified as negatives.

Table.2(a). Performance Metrics Assessment

Model	Precision	Recall	Accuracy	F1
OneR	0.70	0.68	0.69	0.69
Decision Table	0.72	0.70	0.71	0.71
JRIP	0.75	0.73	0.74	0.74
Ridor	0.74	0.72	0.73	0.73
J48	0.80	0.78	0.79	0.79
LMT	0.82	0.80	0.81	0.81
Random Forest	0.85	0.83	0.84	0.84
CART	0.81	0.79	0.80	0.80
MLR	0.78	0.75	0.77	0.76
MLP	0.84	0.82	0.83	0.83
SVM	0.87	0.85	0.86	0.86
LogitBoost	0.83	0.81	0.82	0.82
SMO	0.86	0.84	0.85	0.85
ANN	0.88	0.86	0.87	0.87

Table.2(a). Performance Metrics Assessment

Model	TPR	FPR	TNR	FNR
OneR	0.68	0.15	0.85	0.32
Decision Table	0.70	0.13	0.87	0.30
JRIP	0.73	0.12	0.88	0.27
Ridor	0.72	0.14	0.86	0.28
J48	0.78	0.10	0.90	0.22
LMT	0.80	0.09	0.91	0.20
Random Forest	0.83	0.07	0.93	0.17
CART	0.79	0.09	0.91	0.21
MLR	0.75	0.11	0.89	0.25
MLP	0.82	0.08	0.92	0.18
SVM	0.85	0.06	0.94	0.15
LogitBoost	0.81	0.08	0.92	0.19
SMO	0.84	0.07	0.93	0.16
ANN	0.86	0.05	0.95	0.14

In this analysis, the performance metrics for each model were computed based on three different data splits: training (60%), validation (20%), and testing (20%). Tree-based models, particularly Random Forest and J48, shown higher accuracy, precision, and recall compared to rules-based models, indicating their robustness and ability to capture complex patterns in the data. Among function-based models, SVM and ANN achieved the highest metrics, showcasing their strength in handling non-linear relationships. ANN, with a precision of 0.88 and recall of 0.86, emerged as the top performer, likely due to its deep architecture capable of learning intricate data features. In contrast, rules-based models like OneR and Ridor, while interpretable, showed lower performance metrics, suggesting limited capacity for capturing complex data interactions. Thus, the results indicate that function-based models, followed by tree-based models, provide superior predictive performance for weather forecasting tasks, while rules-based models offer easier interpretability at the cost of lower accuracy.

Table.3. Accuracy for training, testing and validation sets

Model	Training	Testing	Validation
OneR	0.68	0.65	0.64
Decision Table	0.71	0.68	0.67
JRIP	0.73	0.70	0.69
Ridor	0.71	0.68	0.67
J48	0.80	0.76	0.75
LMT	0.82	0.78	0.77
Random Forest	0.85	0.81	0.80
CART	0.79	0.75	0.74
MLR	0.76	0.72	0.71
MLP	0.83	0.79	0.78
SVM	0.87	0.83	0.82
LogitBoost	0.81	0.77	0.76
SMO	0.84	0.80	0.79
ANN	0.88	0.84	0.83

The accuracy values across different models and data splits show varying degrees of performance:

- **Rules-based Models:** These models (OneR, Decision Table, JRIP, Ridor) tend to have lower accuracy compared to tree-based and function-based models. For example, OneR shows the lowest accuracy, indicating it may not handle complex patterns well. The accuracy improves slightly with more sophisticated rules-based models like JRIP.
- **Tree-based Models:** Models like Random Forest and J48 generally perform better than rules-based models. Random Forest, in particular, exhibits high accuracy across all data splits, reflecting its ability to generalize well due to its ensemble nature. LMT and CART also show good performance, though slightly less than Random Forest.
- **Function-based Models:** These models tend to achieve the highest accuracy. For instance, ANN (Artificial Neural Network) achieves the highest accuracy Thus, which is expected due to its capability to model complex, non-linear relationships. SVM also shows strong performance, especially in the testing set, which indicates its effectiveness in creating optimal decision boundaries.

Table.4. Precision for training, testing and validation sets

Model	Training	Testing	Validation
OneR	0.66	0.64	0.63
Decision Table	0.70	0.67	0.66
JRIP	0.72	0.69	0.68
Ridor	0.68	0.65	0.64
J48	0.78	0.74	0.73
LMT	0.81	0.77	0.76
Random Forest	0.84	0.80	0.79
CART	0.76	0.72	0.71
MLR	0.73	0.69	0.68
MLP	0.80	0.76	0.75
SVM	0.85	0.81	0.80
LogitBoost	0.77	0.73	0.72

SMO	0.82	0.78	0.77
ANN	0.87	0.83	0.82

The precision values reflect each model's ability to correctly identify positive instances:

- **Rules-based Models:** Precision is generally lower across rules-based models (OneR, Decision Table, JRIP, Ridor). For example, OneR shows the lowest precision, indicating it struggles with identifying true positives accurately. JRIP and Decision Table perform slightly better but still lag behind more advanced models.
- **Tree-based Models:** These models, such as Random Forest and J48, show improved precision compared to rules-based models. Random Forest stands out with the highest precision, demonstrating its effectiveness in reducing false positives due to its ensemble approach. J48 and LMT also perform well but are slightly less precise than Random Forest.
- **Function-based Models:** Function-based models achieve the highest precision values. SVM and ANN excel in identifying true positives, with ANN achieving the highest precision overall. This high precision reflects ANN's ability to learn complex patterns and minimize false positives effectively. SVM also shows strong precision, confirming its robustness in classifying data accurately.

Thus, function-based models, especially ANN, provide superior precision, followed by tree-based models, while rules-based models exhibit lower precision, indicating less effectiveness in minimizing false positives.

Table.5. Recall for training, testing and validation sets

Model	Training	Testing	Validation
OneR	0.62	0.60	0.59
Decision Table	0.66	0.64	0.63
JRIP	0.69	0.66	0.65
Ridor	0.64	0.61	0.60
J48	0.75	0.72	0.71
LMT	0.78	0.74	0.73
Random Forest	0.81	0.77	0.76
CART	0.70	0.68	0.67
MLR	0.68	0.65	0.64
MLP	0.76	0.72	0.71
SVM	0.82	0.78	0.77
LogitBoost	0.73	0.70	0.69
SMO	0.77	0.73	0.72
ANN	0.85	0.81	0.80

Recall measures the ability of a model to identify all relevant positive instances:

- **Rules-based Models:** These models (OneR, Decision Table, JRIP, Ridor) generally have lower recall values, reflecting their challenges in identifying all true positives. For instance, OneR shows the lowest recall, indicating it misses a significant portion of positive instances. JRIP performs somewhat better but still does not match the recall of more advanced models.
- **Tree-based Models:** Tree-based models exhibit improved recall compared to rules-based models. Random Forest, with the highest recall among tree-based models, indicates its effectiveness in capturing most positive instances. J48 and LMT also perform well, showing better recall than their rules-based counterparts but still fall short of function-based models.
- **Function-based Models:** Function-based models achieve the highest recall values. ANN stands out with the highest recall Thus, demonstrating its capacity to identify the majority of positive instances. SVM also shows strong recall, confirming its ability to effectively capture positive cases. These models, especially ANN, excel in reducing false negatives and ensuring comprehensive identification of positive instances.

Thus, function-based models, particularly ANN, offer superior recall, capturing more true positives, followed by tree-based models. Rules-based models generally show lower recall, indicating limited effectiveness in identifying all relevant positive instances.

Table.6. F-measure for training, testing and validation sets

Model	Training	Testing	Validation
OneR	0.64	0.61	0.60
Decision Table	0.68	0.65	0.64
JRIP	0.71	0.68	0.67
Ridor	0.66	0.63	0.62
J48	0.76	0.72	0.71
LMT	0.79	0.75	0.74
Random Forest	0.83	0.78	0.77
CART	0.73	0.70	0.69
MLR	0.71	0.68	0.67
MLP	0.78	0.74	0.73
SVM	0.85	0.81	0.80
LogitBoost	0.76	0.72	0.71
SMO	0.80	0.76	0.75
ANN	0.86	0.82	0.81

The F-measure, which balances precision and recall, provides a single metric to evaluate the performance of each model:

- **Rules-based Models:** The F-measure is relatively lower for rules-based models (OneR, Decision Table, JRIP, Ridor), reflecting their limited ability to balance precision and recall. For example, OneR has the lowest F-measure, indicating it struggles to effectively manage both false positives and false negatives.
- **Tree-based Models:** Tree-based models show improved F-measure values compared to rules-based models. Random Forest achieves the highest F-measure among tree-based models, indicating it effectively balances precision and recall. J48 and LMT also perform well, but their F-measure values are slightly lower, suggesting some trade-off between precision and recall.
- **Function-based Models:** Function-based models exhibit the highest F-measure values. ANN leads with the highest

F-measure, demonstrating its superior ability to balance precision and recall, ensuring a robust performance across all metrics. SVM also performs strongly, reflecting its effective balance between correctly identifying positives and minimizing false positives and negatives.

Thus, function-based models, especially ANN, show the best F-measure, providing a strong balance of precision and recall. Tree-based models follow, offering good performance but with some trade-offs. Rules-based models generally show lower F-measure, indicating less effective management of precision and recall.

Table.7. Accuracy over various Datasets

Model	Kaggle Weather Forecast	IEEE Weather Forecast
OneR	0.63	0.62
Decision Table	0.66	0.65
JRIP	0.68	0.67
Ridor	0.65	0.64
J48	0.75	0.73
LMT	0.77	0.76
Random Forest	0.80	0.78
CART	0.72	0.70
MLR	0.70	0.68
MLP	0.76	0.74
SVM	0.82	0.80
LogitBoost	0.74	0.72
SMO	0.78	0.76
ANN	0.84	0.82

The accuracy values reflect each model's performance on weather prediction datasets from Kaggle and IEEE:

- **Rules-based Models:** These models (OneR, Decision Table, JRIP, Ridor) exhibit lower accuracy compared to more advanced models. For instance, OneR achieves an accuracy of 0.63 on Kaggle and 0.62 on IEEE, indicating limited capability in handling complex weather patterns. JRIP performs somewhat better but still shows lower accuracy than tree-based and function-based models.
- **Tree-based Models:** Tree-based models, such as Random Forest and J48, demonstrate higher accuracy. Random Forest achieves an accuracy of 0.80 on Kaggle and 0.78 on IEEE, reflecting its strength in capturing complex patterns through its ensemble approach. J48 and LMT also perform well, showing better accuracy than rules-based models but less than function-based models.
- **Function-based Models:** Function-based models, including SVM and ANN, show the highest accuracy across both datasets. ANN achieves the highest accuracy, with 0.84 on Kaggle and 0.82 on IEEE, indicating its superior ability to model intricate weather patterns. SVM also performs well, with accuracy values of 0.82 and 0.80, respectively.

Thus, function-based models, particularly ANN, provide the highest accuracy, reflecting their effectiveness in predicting weather patterns. Tree-based models follow, showing strong performance but with slightly lower accuracy. Rules-based models generally have the lowest accuracy, highlighting their limitations in handling complex weather data.

Table.8. Precision over various Datasets

Model	Kaggle Weather Forecast	IEEE Weather Forecast
OneR	0.60	0.59
Decision Table	0.62	0.61
JRIP	0.65	0.63
Ridor	0.61	0.60
J48	0.73	0.71
LMT	0.76	0.74
Random Forest	0.79	0.77
CART	0.70	0.68
MLR	0.68	0.66
MLP	0.74	0.72
SVM	0.80	0.78
LogitBoost	0.72	0.70
SMO	0.76	0.74
ANN	0.82	0.80

Precision measures the proportion of true positive predictions among all positive predictions made by the model:

- **Rules-based Models:** These models (OneR, Decision Table, JRIP, Ridor) exhibit lower precision values. For example, OneR shows a precision of 0.60 on Kaggle and 0.59 on IEEE, indicating a higher rate of false positives. JRIP performs slightly better but still lags behind more advanced models.
- **Tree-based Models:** Tree-based models such as Random Forest and J48 show improved precision compared to rules-based models. Random Forest achieves a precision of 0.79 on Kaggle and 0.77 on IEEE, reflecting its ability to better identify relevant positive instances. J48 and LMT also perform well, indicating good precision but slightly lower than Random Forest.
- **Function-based Models:** Function-based models demonstrate the highest precision. ANN achieves the highest precision, with 0.82 on Kaggle and 0.80 on IEEE, indicating its effectiveness in minimizing false positives. SVM also performs strongly, with precision values of 0.80 and 0.78, respectively.

Thus, function-based models, especially ANN, provide the highest precision, effectively identifying true positives with fewer false positives. Tree-based models follow, showing solid precision but less than function-based models. Rules-based models generally exhibit lower precision, highlighting their higher rate of false positives in weather prediction tasks.

Table.9. Recall over various Datasets

Model	Kaggle Weather Forecast	IEEE Weather Forecast

OneR	0.58	0.57
Decision Table	0.61	0.60
JRIP	0.64	0.62
Ridor	0.59	0.58
J48	0.72	0.69
LMT	0.75	0.73
Random Forest	0.78	0.76
CART	0.70	0.68
MLR	0.66	0.64
MLP	0.73	0.71
SVM	0.79	0.77
LogitBoost	0.71	0.69
SMO	0.74	0.72
ANN	0.81	0.79

Recall measures the proportion of true positive instances identified by the model out of all actual positive instances:

- **Rules-based Models:** These models (OneR, Decision Table, JRIP, Ridor) exhibit lower recall values. For instance, OneR has a recall of 0.58 on Kaggle and 0.57 on IEEE, indicating that it misses a significant number of positive instances. JRIP performs slightly better but still shows limited effectiveness in capturing all relevant positives compared to more advanced models.
- **Tree-based Models:** Tree-based models show improved recall compared to rules-based models. Random Forest, with a recall of 0.78 on Kaggle and 0.76 on IEEE, effectively identifies a higher proportion of positive instances. J48 and LMT also perform well, reflecting better recall but slightly lower than Random Forest.
- **Function-based Models:** Function-based models achieve the highest recall values. ANN leads with the highest recall, reaching 0.81 on Kaggle and 0.79 on IEEE, demonstrating its strong capability to identify most positive instances. SVM also performs robustly, with recall values of 0.79 and 0.77, respectively.

Thus, function-based models, particularly ANN, provide the highest recall, effectively identifying most positive instances with fewer missed positives. Tree-based models follow with strong performance, while rules-based models generally exhibit lower recall, indicating less comprehensive identification of positive instances.

Table.10. F-Measure over various Datasets

Model	Kaggle Weather Forecast	IEEE Weather Forecast
OneR	0.59	0.58
Decision Table	0.62	0.61
JRIP	0.66	0.64
Ridor	0.60	0.59
J48	0.73	0.71
LMT	0.76	0.74
Random Forest	0.79	0.77
CART	0.71	0.69
MLR	0.67	0.65
MLP	0.74	0.72
SVM	0.80	0.78
LogitBoost	0.72	0.70
SMO	0.75	0.73
ANN	0.82	0.80

The F-measure balances precision and recall, providing a single metric that combines both aspects of model performance:

- **Rules-based Models:** These models (OneR, Decision Table, JRIP, Ridor) show lower F-measure values, reflecting difficulties in achieving a good balance between precision and recall. For example, OneR has an F-measure of 0.59 on Kaggle and 0.58 on IEEE, indicating that it struggles to manage both false positives and false negatives effectively.
- **Tree-based Models:** Tree-based models exhibit better F-measure scores. Random Forest leads with an F-measure of 0.79 on Kaggle and 0.77 on IEEE, demonstrating a strong balance between precision and recall. J48 and LMT also perform well, showing improvements over rules-based models but slightly lower than Random Forest.
- **Function-based Models:** Function-based models achieve the highest F-measure values. ANN, with an F-measure of 0.82 on Kaggle and 0.80 on IEEE, excels in balancing precision and recall. SVM also performs strongly, with F-measure values of 0.80 and 0.78, respectively.

Thus, function-based models, particularly ANN, provide the highest F-measure, effectively balancing precision and recall. Tree-based models follow with robust performance but slightly lower F-measure. Rules-based models generally exhibit lower F-measure, indicating less effective management of both precision and recall.

5. DISCUSSION OF RESULTS

The performance metrics for the various machine learning models—rules-based, tree-based, and function-based—on the Kaggle and IEEE weather prediction datasets reveal significant insights into their efficacy in weather forecasting tasks.

5.1 RULES-BASED MODELS

The rules-based models, including OneR, Decision Table, JRIP, and Ridor, generally exhibit lower performance across all metrics—accuracy, precision, recall, and F-measure. These models often have lower precision, indicating a higher rate of false positives, and lower recall, reflecting their struggle to identify all relevant positive instances. For instance, OneR's accuracy of 0.63 and precision of 0.60 highlight its limitations in distinguishing between positive and negative cases effectively. The lower F-measure values confirm that these models struggle to balance precision and recall, making them less effective for complex weather prediction tasks. Their performance suggests that while they may be simpler and easier to interpret, their limited complexity hinders their ability to model the intricate patterns present in weather data.

5.2 TREE-BASED MODELS

Tree-based models, such as J48, LMT, Random Forest, and CART, demonstrate improved performance over rules-based models. Random Forest, in particular, achieves high accuracy (0.80) and precision (0.79), indicating its effectiveness in handling complex weather data through its ensemble approach. The high recall values (0.78) and F-measure scores (0.79) further illustrate Random Forest's ability to effectively identify positive instances while balancing precision and recall. J48 and LMT also show strong performance, although slightly lower than Random Forest. These models benefit from their ability to capture non-linear relationships and interactions in the data, which contributes to their superior performance compared to simpler rules-based approaches.

5.3 FUNCTION-BASED MODELS

Function-based models, including MLR, MLP, SVM, LogitBoost, SMO, and ANN, exhibit the highest performance across all metrics. ANN, in particular, stands out with the highest accuracy (0.84) and F-measure (0.82), demonstrating its superior ability to learn complex patterns in the data. The high precision (0.82) and recall (0.81) values of ANN confirm its effectiveness in identifying positive instances while minimizing false positives and negatives. SVM also performs strongly, reflecting its robustness in classifying data accurately. These models leverage advanced techniques such as neural networks and support vector machines to achieve superior performance, highlighting their capability to manage the complexities of weather forecasting tasks more effectively than both rules-based and tree-based models.

Thus, the results underscore that function-based models, particularly ANN, provide the most reliable performance for weather prediction tasks, balancing accuracy, precision, recall, and F-measure effectively. Tree-based models offer significant improvements over rules-based models but are outperformed by function-based models in handling complex weather data.

6. INFERENCES

The superior performance of function-based models, particularly Artificial Neural Networks (ANN), suggests that these models are highly effective for complex tasks such as weather prediction. ANN's high accuracy, precision, recall, and F-measure indicate its capacity to capture and learn intricate patterns and relationships within weather data. This capability is crucial for forecasting tasks where accurate prediction and minimal false positives and negatives are essential. The robustness of ANN in managing diverse and complex datasets highlights the advantages of employing advanced machine learning techniques that leverage deep learning and neural network architectures.

Rules-based models, including OneR, Decision Table, JRIP, and Ridor, demonstrate clear limitations in weather prediction tasks. Their lower performance across all metrics—accuracy, precision, recall, and F-measure—suggests that they struggle with the complexity of weather data. These models, which rely on simple rule-based logic, are less capable of handling non-linear relationships and intricate patterns compared to more advanced models. As a result, their use in practical applications where high accuracy and comprehensive prediction are required may be limited. The challenges observed with rules-based models highlight the need for more sophisticated approaches to improve performance in complex forecasting scenarios.

Tree-based models, such as J48, LMT, Random Forest, and CART, show notable improvements over rules-based models. Their ability to handle non-linear relationships and interactions in the data contributes to their enhanced performance in weather prediction tasks. Random Forest, in particular, stands out for its robustness and high performance, reflecting its strength in aggregating multiple decision trees to improve predictive accuracy. Tree-based models offer a middle ground between simplicity and complexity, making them a viable option for tasks requiring a balance of interpretability and performance. However, while they represent an advancement over rules-based models, they are still outperformed by function-based models, which leverage deeper learning techniques for even greater accuracy.

The findings emphasize the need for selecting appropriate models based on the complexity of the task and the dataset. Function-based models, especially ANN, are recommended for applications demanding high accuracy and robust performance in handling complex and dynamic data. Tree-based models provide a strong alternative for cases where interpretability and moderate performance are acceptable. Rules-based models may be suitable for simpler tasks but fall short in scenarios requiring high precision and recall. The insights underscore the importance of model selection and the value of advanced techniques in achieving optimal performance for complex predictive tasks.

Conclusion

The comparative analysis of machine learning models for weather prediction tasks on the Kaggle and IEEE datasets highlights several key findings that shape the conclusion of this study. Function-based models, particularly Artificial Neural Networks (ANN), emerge as the most effective approach for weather forecasting. Their superior performance across all evaluation metrics—accuracy, precision, recall, and F-measure—demonstrates their ability to handle the complexities and intricacies of weather data with remarkable proficiency. ANN's high accuracy of 0.84 and F-measure of 0.82 reflect its capacity to accurately predict weather patterns while effectively managing false positives and negatives. The robust performance of ANN underscores the advantages of leveraging advanced deep learning techniques in achieving high-quality predictions. In contrast, rules-based models, such as OneR, Decision Table, JRIP, and Ridor, exhibit lower performance in comparison to both tree-based and function-based models. Their limited accuracy and lower precision and recall values indicate their struggle with the complexity of weather prediction tasks. These models, which rely on straightforward rule-based logic, are less adept at capturing the non-linear relationships inherent in weather data. Their performance suggests that while they may be simpler to implement, they are less suitable for applications requiring high precision and comprehensive predictive capabilities. Tree-based models, including J48, LMT, Random Forest, and CART, offer a notable improvement over rules-based models. Their ability to manage non-linear data relationships and interactions results in better performance metrics. Random Forest provides a strong balance between performance and complexity, achieving high

accuracy and recall. However, while tree-based models represent a significant advancement over rules-based approaches, they still fall short of the superior performance achieved by function-based models. Thus, this study highlights the importance of selecting the appropriate model based on the complexity of the forecasting task. Function-based models, especially ANN, are recommended for scenarios requiring high accuracy and robust performance. Tree-based models serve as a viable middle ground, while rules-based models are less effective for complex prediction tasks. The findings emphasize the critical role of model sophistication in achieving optimal performance in weather prediction.

REFERENCES

[1] Shi Zhang and Dingwei Wang, "Medium and Long-Term Load Forecasting based on PCA and BP Neural Network Method", *Proceedings of International Conference on Energy and Environmental Technology*, pp. 389-391, 2009.

[2] Z.A. Bashir and M.E. EI-Hawary, "Applying Wavelets to Short-Term Load Forecasting using PSO-Based Neural Networks", *IEEE Transaction on Power Systems*, Vol. 24, No.1, pp. 20-27, 2009.

[3] L.M. Trick, R. Toxopeus and D. Wilson, "The Effects of Visibility Conditions, Traffic Density, and Navigational Challenge on Speed Compensation and Driving Performance in Older Adults", *Accident Analysis and Prevention*, Vol. 42, No. 6, pp. 1661-1671, 2010.

[4] W.S. Ashley, S. Strader, D.C. Dziubla and A. Haberlie, "Driving Blind: Weather-Related Vision Hazards and Fatal Motor Vehicle Crashes", *Bulletin of the American Meteorological Society*, Vol. 96, No. 5, pp. 755-778, 2015

[5] M.M. Ahmed, M. Abdel-Aty, J. Lee and R. Yu, "Real-Time Assessment of Fog-Related Crashes using Airport Weather Data: A Feasibility Analysis", *Accident Analysis and Prevention*, Vol. 72, pp. 309-317, 2014.

[6] A. Vlachogianni, P. Kassomenos, A. Karppinen, S. Karakitsios and J. Kukkonen, "Evaluation of a Multiple Regression Model for the Forecasting of the Concentrations of Nox and Pm10 in Athens and Helsinki", *Science of the Total Environment*, Vol. 409, No. 8, pp. 1559-1571, 2011.

[7] V.R. Thakare and H.M. Baradkar, "Fuzzy System for Maximum Yield from Crops", *Proceedings of National Level Technical Conference on Data Mining*, pp. 4-9, 2013.

[8] Kneale T. Marshall, "Decision Making and Forecasting: With Emphasis on Model Building and Policy Analysis", McGraw-Hill, 1995.

[9] David E. Rumelhart and David Zipser, "Feature Discovery by Competitive Learning", *Cognitive Science*, Vol. 9, No. 1, pp. 75-112, 1985.

[10] Teuvo Kohonen, "An Introduction to Neural Computing", *Neural Networks*, Vol. 1, No. 1, pp. 3-16, 1988.

[11] M.A.R. Suleman and S. Shridevi, "Short-Term Weather Forecasting using Spatial Feature Attention based LSTM Model", *IEEE Access*, Vol. 10, pp. 82456-82468, 2022.

[12] P. Hewage and A. Behera, "Deep Learning-Based Effective Fine-Grained Weather Forecasting Model", *Pattern Analysis and Applications*, Vol. 24, No. 1, pp. 343-366, 2021.

[13] B. Bochenek and M. Figurski, "Day-Ahead Wind Power Forecasting in Poland based on Numerical Weather Prediction", *Energies*, Vol. 14, No. 8, pp. 2164-2173, 2021.